New FFT structures based on the Bruun algorithm

Wu, Yuhang

Published in:
IEEE Transactions on Acoustics, Speech, and Signal Processing

Link to article, DOI:
10.1109/29.45572

Publication date:
1990

Document Version
Publisher's PDF, also known as Version of record

EDCT-III:

$$X_+^{c3}(m) = \cos\frac{(2m+1)\pi}{2N} X^{c3}(m) + \sin\frac{(2m+1)\pi}{2N} X^{s3}(m+1) + \sqrt{\frac{2}{N}}\left[-\sqrt{\frac{1}{2}}x(0)\cos\frac{(2m+1)\pi}{2N} + \left(\sqrt{\frac{1}{2}}-1\right)x(1) + (-1)^m\left(1-\sqrt{\frac{1}{2}}\right)x(N)\cdot\sin\frac{(2m+1)\pi}{2N}\right], \quad m = 0, 1, \cdots, N-1. \tag{24}$$

EDST-II:

$$X_+^{s2}(m) = \cos\frac{m\pi}{N} X^{s2}(m) - \sin\frac{m\pi}{N} X_+^{c2}(m) + \sqrt{\frac{2}{N}} k_m \sin\frac{m\pi}{2N}\left[x(1) - (-1)^m x(N+1)\right], \quad m = 1, 2, \cdots, N. \tag{25}$$

EDST-III:

$$X_+^{s3}(m) = \cos\frac{(2m-1)\pi}{2N} X^{s3}(m) - \sin\frac{(2m-1)\pi}{2N} X^{c3}(m-1) - \sqrt{\frac{2}{N}}\left[-\sqrt{\frac{1}{2}}x(0)\sin\frac{(2m-1)\pi}{2N} + (-1)^m\left(1-\sqrt{\frac{1}{2}}\right)x(N)\cos\frac{(2m-1)\pi}{2N} + (-1)^m\sqrt{\frac{1}{2}}x(N+1)\right], \quad m = 1, 2, \cdots, N. \tag{26}$$

In addition, the representation for EDCT-I [(9) in the above[1]] can be simplified as

$$X_+^{c1}(m) = \cos\frac{m\pi}{N} X^{c1}(m) + \sin\frac{m\pi}{N} X^{s1}(m) + \sqrt{\frac{2}{N}} k_m\left[-\sqrt{\frac{1}{2}}x(0)\cos\frac{m\pi}{N} + \left(\sqrt{\frac{1}{2}}-1\right)x(1) + (-1)^m\left(1-\sqrt{\frac{1}{2}}\right)x(N)\cos\frac{m\pi}{N} + (-1)^m\sqrt{\frac{1}{2}}x(N+1)\right], \quad m = 0, 1, \cdots, N. \tag{27}$$

In conclusion, every odd version in the family of DCT and DST possesses, as would be expected, a similar shift property as its even counterpart. Along with the derivation of generalized convolution properties for versions I and II in the family [2], the basic properties for some important transforms have been realized, thus making it more convenient to use these discrete transforms for digital signal processing.

References

[1] Z. Wang and B. R. Hunt, "The discrete *W* transform," *Appl. Math. Comput.*, vol. 16, pp. 19-48, 1985.
[2] L. Wu, "The convolution-multiplication relation for several discrete sine and cosine transforms" (in Chinese), to be published.

New FFT Structures Based on the Bruun Algorithm

YUHANG WU

***Abstract*—In some signal processing applications, the input data are real. In this case, the Bruun algorithm for computation of the Discrete Fourier Transform (DFT) is attractive. This correspondence offers a pipeline and a recirculated shuffle network implementation of the Bruun algorithm. The parallel pipeline and recirculated FFT structures are implemented based on the modified perfect shuffle network.**

I. Introduction

The Discrete Fourier Transform (DFT) of an N-point sequence $x(k)$ is defined by

$$X(n) = \sum_{k=0}^{N-1} x(k)\omega_N^{nk} \qquad n = 0, 1, 2, \cdots, N-1 \tag{1}$$

where $\omega_N = e^{j2\pi/N}$.

Many methods for efficient computation of the DFT have been developed. In [1], Bruun introduced an FFT algorithm, known as the Bruun algorithm, which implements the DFT as a filter bank. The Bruun algorithm computes the DFT of real data with real arithmetic. Among the FFT algorithms, the Bruun algorithm is the most efficient for DFT of real data. In this correspondence, we will present a pipeline and parallel processor implementation of the Bruun algorithm.

In [2]–[4], the perfect shuffle is used for computation of the FFT. The implementation of the FFT based on the perfect shuffle has an interesting property that makes it possible to use a recirculated network. In recent years, all the implementations of FFT using the perfect shuffle are concerned with the classical radix-2 FFT algorithm. We will show how the Bruun algorithm can be implemented using the modified perfect shuffle network.

II. Review of the Bruun Algorithm Fundamentals

In the Bruun algorithm, the DFT $X(n)$ of a sequence $x(k)$ is evaluated by finding the Z-transform of the sequence $x(k)$ at N uniformly spaced points around the unit circle [1]. Fig. 1 shows the signal flow graph of the Bruun algorithm. In order to derive the new FFT structures based on the Bruun algorithm, we need to modify the signal flow graph (Fig. 1). The modified signal flow graph is shown in Fig. 2. Here we have added the dot and dash lines with a multiplicative factor zero into the figure. By careful examination of the signal flow graph (Fig. 2), it can be seen that the basic computation at the kth stage, except the last stage in the flow graph (Fig. 2), is shown in Fig. 3. The equations represented by this flow graph are of the form

$$X_i^k = X_i^{k-1} + F_j X_{i+N/2^{k+1}}^{k-1} + X_{i+N/2^k}^{k-1} \tag{2}$$

$$X_{i+N/2^k}^k = \begin{cases} X_i^{k-1} - X_{i+N/2^k}^{k-1} & j = 0 \\ X_i^{k-1} - F_j X_{i+N/2^{k+1}}^{k-1} + X_{i+N/2^k}^{k-1} & j \neq 0 \end{cases} \tag{3}$$

Manuscript received June 9, 1987; revised May 5, 1989.

The author is with the Electronics Laboratory, Electronics Institute, Technical University of Denmark, DK-2800 Lyngby, Denmark.

IEEE Log Number 8931710.

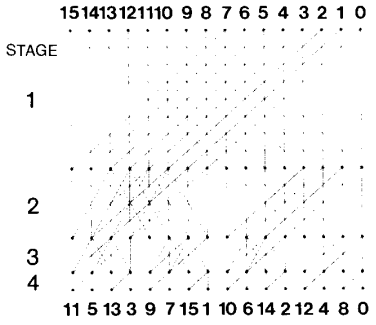


Fig. 1. The signal flow graph of the Bruun algorithm for the case of $N = 16$.

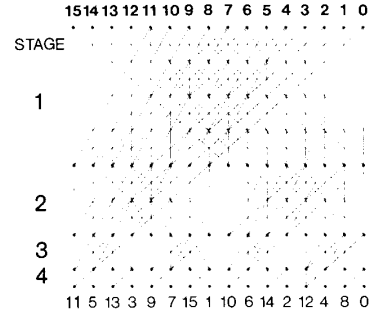
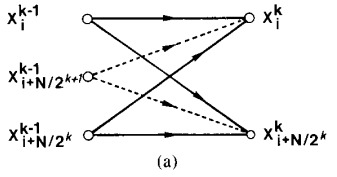


Fig. 2. The modified signal flow graph of a 16-point FFT. The dot and dash lines represent the arrows which have a multiplicative factor zero.



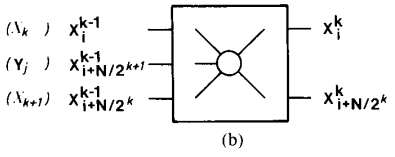


Fig. 3. (a) The signal flow graph of the basic computation in the Bruun algorithm. (b) The symbol of the Bruun algorithm butterfly.

$$k = 1, 2, 3, \cdots, \log N - 1$$

$$i = 0, 1, 2, \cdots, N - 1$$

$$i \text{ MOD } N/2^k = i \text{ MOD } N/2^{k-1}$$

$$j = \text{truncation of } i \cdot 2^{k-1}/N$$

where X_i^k is the signal value represented by the ith node at the kth stage (numbered from 0 to $N - 1$ from right to left) in Fig. 2 and F_j is the filter coefficient represented by dash or dot and dash lines in Fig. 2. We will call this basic computation the Bruun algorithm butterfly. The computations performed at the last stage are of the form

$$X_i^{\log N} = X_i^{\log N - 1} - ZX_{i+1}^{\log N - 1} \tag{4}$$

$$X_{i+1}^{\log N} = X_i^{\log N - 1} - Z^* X_{i+1}^{\log N - 1} \tag{5}$$

where $Z = e^{-j2\pi n/N}$, $n = 1, 2, \cdots, \frac{1}{2}N - 1$; Z^* denote the complex conjugate of Z. Note: n is not equal to i, $Z = -1$ and $Z^* = 1$ in the case of $n = 0$.

TABLE I
THE REAL DFT ALGORITHM

```
1. FOR  k=1  TO  log N-1  DO
2.   BEGIN
3.     COMPUTE_F_j    /* j = 1, 2, ..., N/2 */
                      /* F_j are the DFT filter coefficients */
4.     FOR  i=0  TO  N-1  DO
5.       BEGIN
6.         j = TRUNC( i*2^(k-1) /N )
7.         IF  ( i MOD N/2^k ) = ( i MOD N/2^(k-1) )
             THEN BEGIN
8.                 X_i^k = X_i^(k-1) + F_j * X_(i+N/2^(k+1))^(k-1) + X_(i+N/2^k)^(k-1)
9.               IF  j=0  THEN  X_(i+N/2^k)^k = X_i^(k-1) - X_(i+N/2^k)^(k-1)
                          ELSE  X_(i+N/2^k)^k = X_i^(k-1) - F_j * X_(i+N/2^(k+1))^(k-1) + X_(i+N/2^k)^(k-1)
10.        END
11.      END
12.  END
13. X_0^(log N) = X_0^(log N-1) + X_1^(log N-1)
14. X_1^(log N) = X_0^(log N-1) - X_1^(log N-1)
15. FOR  i=2  TO  N-1  STEP  2  DO
16.  BEGIN
17.    COMPUTE_INDEX_n  /* n is the index of DFT X(n) */
18.    Re Z = Cos(2πn/N)
19.    Im Z = Sin(2πn/N)
20.    Im X_i^(log N) = Im Z*X_i^(log N-1)      /* Imaginary part of X(n) */
21.    X_i^(log N) = X_i^(log N-1) - Re Z*X_(i+1)^(log N-1)   /* Real part of X(n) */
22.    Im X_(i+1)^(log N) = -Im X_i^(log N)     /* Imaginary part of X(N-n) */
23.    X_(i+1)^(log N) = X_i^(log N)            /* Real part of X(N-n) */
24.  END
25. PERMUTATION_X_i
```

III. IMPLEMENTATION

As already mentioned, the Bruun algorithm is most efficient when the input data are real. We will concentrate on the case of real DFT throughout this correspondence. In Table I, the algorithm for computing real DFT is given.

The statements 8 and 9 in the table perform a butterfly operation defined by (2) and (3) in Section II. Only one multiplier is necessary for implementing the butterfly compared to four in the classical FFT butterfly. The statements 13–24 in Table I perform the computations of the last stage shown in Section II. Since the complex conjugate operation is just a simple sign inversion of the imaginary part of the complex number, only two real multipliers are necessary for implementing this stage.

A. The Serial Pipeline Implementation

The serial pipeline structure of the Bruun algorithm is derived as shown in Fig. 4. In each stage of the pipeline network, the output X_{k+1} of the butterfly block performing the butterfly operation is connected to the input of a shift register of an appropriate length. The register output is connected to the first input x_k of the butterfly block. The other output X_k of the butterfly block is connected to

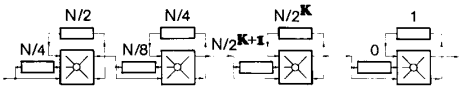


Fig. 4. The serial pipeline implementation of the Bruun algorithm. The numbers near the shift register blocks indicate the lengths of the registers.

the next stage. The second input x_{k+1} of the butterfly block is connected to the previous stage, while the remaining input y_j is connected to the previous stage via another shift register which is half the length of the first one.

Each stage handles the computations associated with a single value of the loop index k in Table I. The operation of the kth stage is as follows. First, the input data pass unchanged into both the shift registers of the lengths $N/2^k$ and $N/2^{k+1}$. Then the incoming data are passed into the shift register of the length $N/2^{k+1}$ and are combined with the data in the two shift registers with the delays $N/2^{k+1}$ and $N/2^k$. One output from the block performing the butterfly operation is sent back into the shift register of the length $N/2^k$, while another is sent to the next stage. When all $N/2^k$ butterfly operations have been completed, the results in the shift register of the length $N/2^k$ are allowed to pass through the block to the next stage, while the next $N/2^k$ point input data are loaded into the two shift registers. The rightmost stage performs the computations of the last stage, i.e., statements 13–24 in Table I. The last statement in Table I performs the permutation of the output values X_i. This procedure, however, cannot be performed on the pipeline structure described above. Note that the classical pipeline FFT produces its output values in bit-reversed order. This is not the case in the structure shown above. If output of natural order is desired, a simple network should be attached to the output of the pipeline FFT network.

B. The Parallel Pipeline Implementation

The parallel pipeline implementation of the Bruun FFT algorithm provides one butterfly block for each butterfly operation in Table I. Fig. 5 shows the parallel pipeline network in the case of $N = 16$. It has $\log N - 1$ stages of $N/2$ butterfly blocks each plus the last stage. Each stage of $N/2$ butterfly blocks performs the entire iterations of the k loop in Table I. The last stage performs the computations from statement 13 to statement 24 in Table I. The input of the network is in the natural order, but the output is not in the bit-reversed order as the classical FFT. A simple network with the wire exchanging may be added to place output in the natural order, if desired. If we number the blocks in Fig. 5 from top to bottom, then the interconnections may be obtained from the following: the block $[i]$ in the kth stage is connected to the blocks $[i/2]$, $[i/2] + N/8$, and $[i/2] + N/4$ in the previous stage. The $[\ ]$ denotes the biggest integer that is less than or equal to the enclosed quantity. The interconnection between the last stage and the last-but-one stage is as follows: the block $[i]$ in the last stage is connected to the two blocks: $[i/2]$ and $[i/2] + N/4$ in the last-but-one stage.

C. The Recirculated Implementation

From [2] and [3], it is known that the recirculated implementation of the classical radix-2 FFT based on the perfect shuffle connection is obtained by connecting the output of the butterflies to the input via a perfect shuffle connection. In order to derive the recirculated implementation of the Bruun algorithm, the butterfly block of Fig. 3 must be modified to perform both the butterfly operation defined by (2) and (3), and the operations of the last stage defined by (4) and (5). Such a butterfly block is associated with three real value input, two complex value output, and the coefficients input. Since the imaginary part of the output has a value only at the end of the entire FFT computation, we are able to deal only with the real part of the output. We now denote two of the input

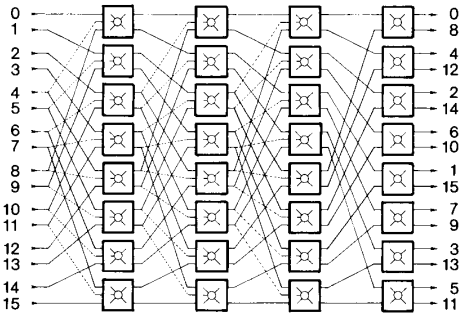


Fig. 5. The parallel pipeline implementation of the Bruun algorithm for $N = 16$.

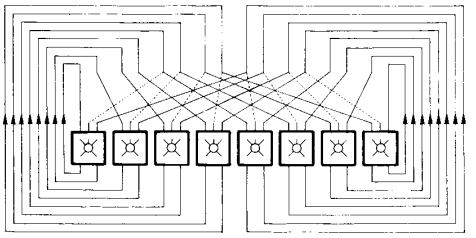


Fig. 6. The recirculated implementation of the Bruun algorithm using modified shuffle connection.

of the butterfly corresponding to the input x_i, $x_{i+N/2^k}$ in Fig. 3 as x_k, x_{k+1}, and the third input as y_j, where k is an even number in range $0 \le k \le N-1$, j is an integer number in range $0 \le j \le N/2 - 1$. Similarly, we denote the two output as X_k and X_{k+1}. We modify the perfect shuffle connection by adding the connection for input y_j. The modified connection scheme can be described by the following equations:

$$\alpha(i) = \begin{cases} 2i & 0 \le i \le \dfrac{N}{2} - 1 \\ 2i + 1 - N & \dfrac{N}{2} \le i \le N - 1 \end{cases} \quad \text{for } x_i \qquad (6)$$

$$\beta(i) = i - \frac{N}{2} \qquad \frac{N}{4} \le i \le \frac{3N}{4} - 1 \quad \text{for } y_i. \qquad (7)$$

A connection is provided from the output X_i of the previous stage to the input x_m of the stage if and only if $m = \alpha(i)$. Similarly, a connection is provided from X_i to y_j if and only if $j = \beta(i)$. Following this scheme, we obtain that for a particular butterfly block containing x_k, x_{k+1}, and $y_{k/2}$, if x_k and x_{k+1} are connected to $X_{\alpha^{-1}(k)}$ and $X_{\alpha^{-1}(k+1)}$, respectively, then $y_{k/2}$ must be connected to $X_{(\alpha^{-1}(k)+\alpha^{-1}(k+1))/2}$, i.e., input y_j is connected to the output of the previous stage with the index which is the middle index of the first two.

It has been shown [6] that the full Bruun algorithm computation can be performed by recycling data through a simple stage consisting of the data propagation along the above connection scheme, followed by the parallel execution of the $N/2$ modified butterfly operations. The different coefficients are applied to the butterfly operations for each time. Such an implementation is shown in Fig. 6.

IV. Conclusion

We have presented the serial and parallel pipeline structures for the implementation of the Bruun algorithm. Furthermore, the recirculated network for implementing the Bruun algorithm is shown.

References

[1] G. Bruun, "Z-transform DFT filters and FFT's," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 56–63, Feb. 1978.
[2] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.
[3] D. S. Parker, "Notes on shuffle/exchange-type switching networks," *IEEE Trans. Comput.*, vol. C-29, pp. 213–222, Mar. 1980.
[4] G. Bongiovanni, "Two VLSI structure for the discrete Fourier transform," *IEEE Trans. Comput.*, vol. C-32, pp. 756–760, Aug. 1983.
[5] C. D. Thompson, "Fourier transform in VLSI," *IEEE Trans. Comput.*, vol. C-32, pp. 1047–1057, Nov. 1983.
[6] Y. Wu, "A comparison of the FFT structures in VLSI," M.Sc. thesis, Electronics Lab., Tech. Univ., Denmark, Aug. 1985.

Performance Analysis of Kimura and Honoki's Hybrid Approach to 2-D Spectral Estimation

XIAN-DA ZHANG AND DA-YONG CUI

Abstract—**More recently, Kimura and Honoki [1] have proposed a hybrid approach to high-resolution 2-D spectral estimation. However, they have not done performance analysis of the final spectrum estimate, and have only conjectured that the final estimate coincides with the true ME estimate for the case of cyclic and skew-cyclic Toeplitz $R(l)$. In this correspondence, we fill the above gap in performance analysis so as to provide a "theoretical" and "practical" solution, and present a counterexample to show Kimura and Honoki's conjecture is not true.**

I. Introduction

The techniques of 2-D maximum entropy (ME) spectral estimation have been recently studied by many researchers. More recently, Kimura and Honoki [1] have proposed a hybrid approach that consists of two steps. The first step is a usual multichannel 1-D ME estimation in the "time" domain. In the second step, the sequence of interchannel cross spectrum obtained in the first step is extended by the ME method in the "frequency" domain. Their interesting approach is practical and attractive since it fully utilizes the high-resolution performance of the ME estimate for 1-D signals, and is able to avoid the highly nonlinear nature of the ME estimation for 2-D signals.

However, some important theoretical problems associated with the estimation algorithm remain to be solved. Especially, Kimura and Honoki have not analyzed the performance of their final spectrum estimate. They have only conjectured that the final spectrum estimate coincides with the true ME estimate for the case of cyclic and skew-cyclic Toeplitz $R(l)$.

In this correspondence, we fill this gap in the performance analysis, and present a counterexample of Kimura and Honoki's conjecture. Our work provides a "theoretical" and "practical" solution to the performance analysis of the final spectrum estimate given by Kimura and Honoki's hybrid approach.

II. Kimura and Honoki's Hybrid Approach

For the convenience of our subsequent performance analysis, we begin by recalling briefly Kimura and Honoki's hybrid approach.

Consider a 2-D homogeneous random field $\{x(t, s)\}$ with zero mean. Let Ω be a rectangular region

$$\Omega = \{(l, k): -L \le l \le L, -K \le k \le K\} \tag{1}$$

and let $r(l, k)$ denote the covariance

$$r(l, k) = E[x(t + l, s + t)x(t, s)] \quad (l, k) \in \Omega \tag{2}$$

where $r(l, k) = r(-l, -k)$ since $\{x(t, s)\}$ is assumed to be homogeneous.

Define the covariance matrix

$$R(l) = \begin{bmatrix} r(l, 0) & r(l, -1) & \cdots & r(l, -K) \\ r(l, 1) & r(l, 0) & \cdots & r(l, -K+1) \\ \vdots & \vdots & & \vdots \\ r(l, K) & r(l, K-1) & \cdots & r(l, 0) \end{bmatrix}. \tag{3}$$

Kimura and Honoki's hybrid approach to high-resolution 2-D spectral estimation consists of two steps.

The first step is to solve the Yule–Walker equation in the "time" domain:

$$[IA_1 \cdots A_L]\Gamma(L) = [\Lambda_L 0 \cdots 0] \tag{4}$$

where $\Gamma(L)$ is the blocked Toeplitz matrix defined by

$$\Gamma(L) = \begin{bmatrix} R(0) & R(1) & \cdots & R(L) \\ R(-1) & R(0) & \cdots & R(L-1) \\ \vdots & \vdots & & \vdots \\ R(-L) & R(-L+1) & \cdots & R(0) \end{bmatrix}. \tag{5}$$

Then, the ME estimate of the spectrum $\Pi(\omega_1) = \Sigma_{l=-\infty}^{\infty} R(l) \exp(-j\omega_1 l)$ for the multichannel 1-D process $X(t) = [x(t, 0), x(t, 1), \cdots, x(t, K)]$ is given by

$$\Pi(\omega_1) = \left[A(\exp(-j\omega_1 l))\right]^{-1} \Lambda_L \left[A(\exp(-j\omega_1 l))\right]^{-T} \tag{6}$$

where

$$A(z_1) = I + A_1 z_1^{-1} + \cdots + A_L z_1^{-L}. \tag{7}$$

The second step is to solve the Yule–Walker equation in the "frequency" domain:

$$[1 a_1(\omega_1) \cdots a_K(\omega_1)]\hat{\Pi}_a(\omega_1) = [\alpha(\omega_1) 0 \cdots 0]. \tag{8}$$

The final spectrum estimate $\hat{P}(\omega_1, \omega_2)$ is given by

$$\hat{P}(\omega_1, \omega_2) = \frac{\alpha(\omega_1)}{\left|a(\omega_1; \exp(j\omega_2))\right|^2} \tag{9}$$

where $a(\omega_1; z_2) = 1 + a_1(\omega_1) z_2^{-1} + \cdots + a_K(\omega_1) z_2^{-K}$.

Define the two matrices M_+ and M_- by

$$M_+ = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

$$M_- = \begin{bmatrix} 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{10}$$

Manuscript received April 23, 1988; revised March 13, 1989. This work was supported in part by the Science Foundation of Aeronautics.

The authors are with Changcheng Institute of Metrology and Measurement (CIMM), P.O. Box 1066, Beijing 100095, China.

IEEE Log Number 8931712.